Q1. Can you create a programme or function that employs both positive and negative indexing? Is there any repercussion if you do so?

A=’jasimsnd’

for i in range(len(A)):

print(A[i])

i+=1

print(A[-i])

program is running fine so I don’t see any repercussions as of now

Q2. What is the most effective way of starting with 1,000 elements in a Python list? Assume that all elements should be set to the same value.

I assume that we have to assign 1000 elements from a python list to same value:

Lis=[,] #list with 1000 elements

for i in range(len(Lis)):

Lis[i]= <some\_value>

Q3. How do you slice a list to get any other part while missing the rest? (For example, suppose you want to make a new list with the elements first, third, fifth, seventh, and so on.)

I will give step size as 2. The program will be as follows

N=100

for i in range(0,N,2):

print(i)

Q4. Explain the distinctions between indexing and slicing.

Indexing means referring an element with its index position and slicing means cutting or sub setting the element.

Q5. What happens if one of the slicing expression's indexes is out of range?

It will just give output for the string that is inside element, or we will receive the output for the indexes that belong to the element

Q6. If you pass a list to a function, and if you want the function to be able to change the values of the list—so that the list is different after the function returns—what action should you avoid?

These are some things that I will avoid:

1. I will not specify the length of the list inside the function
2. I will not name the function with any python’s keywords

Q7. What is the concept of an unbalanced matrix?

Unbalanced matrix means that the matrix is not a square matrix i.e it is a matrix where m !=n where m is number of rows and n is number of columns in the matrix.

Q8. Why is it necessary to use either list comprehension or a loop to create arbitrarily large matrices?

It is because they are both are easier to understand and works very faster with the data provided.